

Broker - Les logs du module WebUI

Sommaire

Chargement des broks initiaux par un regenerator (créateur d'objets des modules de broker) et vérifier que c'est bien la même configuration charger entre les regenerators / Scheduler / Arbiter

Quand un module de broker avec un regenerator charge une nouvelle configuration :

Quand un module de broker avec un regenerator rejette une configuration :

Quand un module de broker avec un regenerator fini de charger une configuration :

Temps de locks trop long entre la consommation des Broks (Ecriture) et les requêtes des utilisateurs (lecture)

Gestion des broks

Information sur l'absorption des broks

Statistiques sur un traitement

Nature des broks traités

L'absorption des broks a pris du retard

Le mode de rattrapage pour récupérer les broks en retard s'active

Le mode rattrapage a suffisamment de broks à traiter

Après avoir traiter des broks, il en reste encore trop en attente

Demande des broks initiaux lors du redémarrage d'un module externe du Broker

Les logs du module MongoDB

Erreurs

La configuration des Graphite backends

Port du Graphite backend invalide

Adresse du Graphite backend vide

Backend mal formé

Protocole du Backend invalide

Les logs de communication avec Graphite

Impossible de joindre le serveur Graphite

Pas de serveur Graphite trouvé

Les log des appels de performances

log de performance de la liste

Les logs du module webui

Création des index en base de données au démarrage

Cas d'erreur

Erreurs lors du lancement du module WebUI

Le port de la WebUI est déjà ouvert

Erreurs issues d'un problème de changement dans le code de la WebUI (problème de DEV), ou d'une installation

cassée si chez un client

Erreurs de paramétrage

Chargement des broks initiaux par un regenerator (créateur d'objets des modules de broker) et vérifier que c'est bien la même configuration charger entre les regenerators / Scheduler / Arbiter

Les logs suivants permettent de suivre le chargement de la configuration de supervision entre l'Arbiter les Schedulers jusqu'aux interfaces : webui / livestatus / livedata

Il existe 2 types d'identifiants de configuration (représentation de la configuration)

- **configuration-uuid**: uuid de configuration totale générée par l'Arbiter
- **shard-id**: id de la partie de configuration géré par un Scheduler

Quand un module de broker avec un regenerator charge une nouvelle configuration :

```
[ YYYY-MM-DD HH:MM:SS ] INFO : [ WebUI-name ] [ REGENERATOR ] [ scheduler=scheduler_name ] Creating new configuration for [shard_id=s hard_id , scheduler= scheduler_name , configuration_uuid= configuration_uuid , arbiter= arbiter_name , architecture= architecture_name , date=creation_date , active= active ]
```

- **shard_id**: id de la partie de configuration gérée par le Scheduler (unique par Scheduler)
- **scheduler_name**: nom du Scheduler qui gère cette partie de la configuration
- **configuration_uuid**: uuid crée lors du démarrage de l'Arbiter qui correspond donc à l'id de la configuration gérée par l'Arbiter
- **creation_date**: date du démarrage de l'Arbiter
- **arbiter_name**: nom de l'Arbiter qui a créé cette configuration
- **architecture_name**: nom de l'architecture, unique par installation de Shinken
- **active**: Est a False si le Scheduler est pas actif (en attente d'une configuration)

Exemple Log Broker - module WebUI3 chargement de la nouvelle configuration

```
[2022-01-26 13:01:43] INFO : [ WebUI3 ] [ REGENERATOR ] [ scheduler=scheduler-dev2 ] Creating new configuration for [shard_id=256, scheduler=scheduler-dev2, configuration_uuid=a549db11b51d4aeb8548b43b088112e9, arbiter=arbiter-dev2, architecture=Shinken-dev2, date=26-01-2022 13:00:28, active=True]
```

Quand un module de broker avec un regenerator rejette une configuration :

Dans le cas où la configuration d'un Scheduler est déjà gérée par un regenerator (cas qui arrive si par exemple un module crash) on redemande les broks initiaux. Tous les modules vont recevoir la nouvelle configuration, mais ceux qui la gèrent déjà, ne vont pas la recharger et vont logger :

```
[YYYY-MM-DD HH:MM:SS] INFO : [WebUI-name] [ CONFIGURATION ] [ NEW ] [ REGENERATOR ] No need to reload the configuration part because I already handle it [shard_id=shard_id, scheduler=scheduler_name, configuration_uuid=configuration_uuid, arbiter=arbiter_name, architecture=architecture_name, date=creation_date, active=active]
```

- **shard_id:** id de la partie de configuration gérée par le Scheduler (unique par Scheduler)
- **scheduler_name:** nom du Scheduler qui gère cette partie de la configuration
- **configuration_uuid:** uuid créée lors du démarrage de l'Arbiter qui correspond donc à l'id de la configuration gérée par l'Arbiter
- **creation_date:** date du démarrage de l'Arbiter
- **arbiter_name:** nom de l'Arbiter qui a créé cette configuration
- **architecture_name:** nom de l'architecture, unique par installation de Shinken
- **active:** Est à False si le Scheduler est pas actif (en attente d'une configuration)

Exemple Log Broker - module WebUI3 chargement de la nouvelle configuration

```
[YYYY-MM-DD HH:MM:SS] WARNING: [WebUI3] [ CONFIGURATION ] [ NEW ] [ REGENERATOR ] No need to reload the configuration part because I already handle it [shard_id=256, scheduler=scheduler-dev2, configuration_uuid=a549db11b51d4aeb8548b43b088112e9, arbiter=arbiter-dev2, architecture=Shinken-dev2, date=26-01-2022 13:00:28, active=True]
```

Quand un module de broker avec un regenerator fini de charger une configuration :

Quand un Scheduler a fini d'envoyer une configuration, le regenerator charge cette configuration les log suivant montre les étapes de ce chargement.

Le début du chargement est montré par le log suivant :

Exemple Log Broker - module WebUI3 chargement de la nouvelle configuration

```
[YYYY-MM-DD HH:MM:SS] INFO : [ WebUI3 ] [ REGENERATOR ] [ scheduler=scheduler-dev2 ] [ LOADING ] Loading configuration part : [shard_id=256, scheduler=scheduler-dev2, configuration_uuid=a549db11b51d4aeb8548b43b088112e9, arbiter=arbiter-dev2, architecture=Shinken-dev2, date=26-01-2022 13:00:28, active=True]
```

Les différentes étapes avec les logs suivants :

Exemple Log Broker - module WebUI3 chargement de la nouvelle configuration

```
[2022-01-26 13:01:43] INFO : [ WebUI3 ] [ REGENERATOR ] [ scheduler=scheduler-dev2 ] [ LOADING ]
[ PERF ] Merging incoming hostgroup with already existing ones ----- : 0.00
[2022-01-26 13:01:43] INFO : [ WebUI3 ] [ REGENERATOR ] [ scheduler=scheduler-dev2 ] [ LOADING ]
[ PERF ] Linking hosts => hostgroups / command / timeperiod / contacts ----- : 0.00
[2022-01-26 13:01:43] INFO : [ WebUI3 ] [ REGENERATOR ] [ scheduler=scheduler-dev2 ] [ LOADING ]
[ PERF ] Merging incoming service groups with already existing ones ----- : 0.00
[2022-01-26 13:01:43] INFO : [ WebUI3 ] [ REGENERATOR ] [ scheduler=scheduler-dev2 ] [ LOADING ]
[ PERF ] Linking services => host / servicegroups / command / timeperiod / contact - : 0.00
[2022-01-26 13:01:43] INFO : [ WebUI3 ] [ REGENERATOR ] [ scheduler=scheduler-dev2 ] [ LOADING ]
[ PERF ] Linking service groups => services ----- : 0.00
[2022-01-26 13:01:43] INFO : [ WebUI3 ] [ REGENERATOR ] [ scheduler=scheduler-dev2 ] [ LOADING ]
[ PERF ] Linking hostgroups => hosts ----- : 0.00
[2022-01-26 13:01:43] INFO : [ WebUI3 ] [ REGENERATOR ] [ scheduler=scheduler-dev2 ] [ LOADING ]
[ PERF ] Build realm list ----- : 0.00
[2022-01-26 13:01:43] INFO : [ WebUI3 ] [ REGENERATOR ] [ scheduler=scheduler-dev2 ] [ LOADING ]
[ PERF ] Linking host => host & check dependencies ( parents / childs ) ----- : 0.00
[2022-01-26 13:01:43] INFO : [ WebUI3 ] [ REGENERATOR ] [ scheduler=scheduler-dev2 ] [ LOADING ]
[ PERF ] Linking checks => host & check dependencies ( parents / childs ) ----- : 0.00
[2022-01-26 13:01:43] INFO : [ WebUI3 ] [ REGENERATOR ] [ scheduler=scheduler-dev2 ] [ LOADING ]
[ PERF ] Linking timeperiod => excluded timeperiod ----- : 0.00
[2022-01-26 13:01:43] INFO : [ WebUI3 ] [ REGENERATOR ] [ scheduler=scheduler-dev2 ] [ LOADING ]
[ PERF ] Merging incoming contactgroups with already existing ones ----- : 0.00
[2022-01-26 13:01:43] INFO : [ WebUI3 ] [ REGENERATOR ] [ scheduler=scheduler-dev2 ] [ LOADING ]
[ PERF ] Linking contactgroups => contacts ----- : 0.00
```

Puis lors que le chargement est fini l'on donne la taille de la configuration total chargé dans le regenerator et le temps pris pour ce chargement.

Exemple Log Broker - module WebUI3 chargement de la nouvelle configuration

```
[2022-01-26 13:01:43] INFO : [ WebUI3 ] [ REGENERATOR ] [ scheduler=scheduler-dev2 ] [ LOADING ]
Configuration size
[2022-01-26 13:01:43] INFO : [ WebUI3 ] [ REGENERATOR ] [ scheduler=scheduler-dev2 ] [ LOADING ]
- hosts ----- : 1
[2022-01-26 13:01:43] INFO : [ WebUI3 ] [ REGENERATOR ] [ scheduler=scheduler-dev2 ] [ LOADING ]
- checks ----- : 27
[2022-01-26 13:01:43] INFO : [ WebUI3 ] [ REGENERATOR ] [ scheduler=scheduler-dev2 ] [ LOADING ]
- contacts ----- : 1
[2022-01-26 13:01:43] INFO : [ WebUI3 ] [ REGENERATOR ] [ scheduler=scheduler-dev2 ] [ LOADING ]
- notificationways --- : 1
[2022-01-26 13:01:43] INFO : [ WebUI3 ] [ REGENERATOR ] [ scheduler=scheduler-dev2 ] [ LOADING ]
- hostgroups ----- : 0
[2022-01-26 13:01:43] INFO : [ WebUI3 ] [ REGENERATOR ] [ scheduler=scheduler-dev2 ] [ LOADING ]
- servicegroups ----- : 0
[2022-01-26 13:01:43] INFO : [ WebUI3 ] [ REGENERATOR ] [ scheduler=scheduler-dev2 ] [ LOADING ]
- contactgroups ----- : 1
[2022-01-26 13:01:43] INFO : [ WebUI3 ] [ REGENERATOR ] [ scheduler=scheduler-dev2 ] [ LOADING ]
- timeperiods ----- : 1
[2022-01-26 13:01:43] INFO : [ WebUI3 ] [ REGENERATOR ] [ scheduler=scheduler-dev2 ] [ LOADING ]
- commands ----- : 219
[2022-01-26 13:01:43] INFO : [ WebUI3 ] [ REGENERATOR ] [ scheduler=scheduler-dev2 ] [ LOADING ]
- realms ----- : 1
[2022-01-26 13:01:43] INFO : [ WebUI3 ] [ REGENERATOR ] [ scheduler=scheduler-dev2 ] [ LOADING ]
The configuration with shard_id=256 was fully load in 0.004s
```

Il y aura une série de log comme ca pour chaque Scheduler contactés par le Broker.

Temps de locks trop long entre la consommation des Broks (Ecriture) et les requêtes des utilisateurs (lecture)

Actuellement on ne sait pas consommer les broks et répondre aux requêtes de l'interface de Visualisation en même temps. On a donc une concurrence entre deux parties:

- Récupération, consommation des broks depuis le broker et mise à jour des hôtes/checks/clusters (et tous les autres objets) depuis les informations des broks
- Réponses aux requêtes de l'interface de Visualisation (*parcours des hôtes, checks, clusters ...*)

Un des principaux risques est une famine d'un des deux groupes d'actions:

- Si on ne fait qu'avaler des broks et ne jamais répondre à l'interface, ceci va poser problème
- Symétriquement, si on ne fait que répondre aux utilisateurs, et jamais avaler des broks, on va avoir des informations périmées, voir, on ne finira jamais de consommer de nouvelles configurations

Le gestionnaire de lock essaie de partager au mieux le temps d'exécution entre les deux groupes, en cas de forte charge, des logs vont remonter les lenteurs observées.

Quand on a trop de requêtes de lectures, et qu'elles ne rendent pas la main pendant plus de 30 sec aux broks, on aura un log suivant (*Brok BLOQUE par les requêtes*):

```
ERROR: [ ITEMS ACCESS ORDONNANCER ] [ LONG LOCK ] Broks management are waiting (1 thread) since 30s (> log error limit=30s) because HTTP requests (20 threads) has the LOCK
```

Quand on a trop de consommation de Broks, et que les requêtes sont bloquées (*Requêtes de l'interface BLOQUÉES par les Broks*)

```
ERROR: [ ITEMS ACCESS ORDONNANCER ] [ LONG LOCK ] HTTP requests are waiting (5 threads) since 30s (> log error limit=30s) because Broks management (1 thread) has the LOCK
```

Quand les requêtes en lecture mettent trop de temps à rendre la main au consommateur de Broks et que d'autres requêtes en lecture attendent de pouvoir s'exécuter depuis trop longtemps :

```
ERROR: [ ITEMS ACCESS ORDONNANCER ] [ LONG LOCK ] Still have 9 running tasks ongoing (HTTP requests). => ( 1 ) Broks management and then ( 11 ) HTTP requests are waiting since 30s (>= log error limit:30s)
```

Quand la consommation de Broks met trop de temps à rendre la main pour la gestion de requêtes en lecture, et que d'autres consommateurs attendent de s'exécuter depuis trop longtemps (*cas théorique, n'est pas supposé survenir en fonctionnement normal*) :

```
ERROR: [ ITEMS ACCESS ORDONNANCER ] [ LONG LOCK ] Still have 1 running tasks ongoing (Broks management). => ( 12 ) HTTP requests then ( 1 ) Broks management are waiting since 30s (>= log error limit:30s)
```

Gestion des broks

Information sur l'absorption des broks

Statistiques sur un traitement

Des broks ont été traités, affichage de statistiques :

- nombre de **broks** traités
- temps d'attente du premier **brok set**
- nombre de **brok set** en retard récupérés, et le temps que ça a pris de les récupérer
- temps passé à désérialiser les **broks**
- temps d'attente du lock avant de traiter les **broks**
- temps passé pour traiter les **broks**
- temps total

```
[YYYY-MM-DD HH:MM:SS] INFO : [ WebUI-name ] [ MANAGE BROKS ] [PERF] [ XXXX broks ] [ wait and get first set on queue=X.XXXs ] [ get 0 late sets on=X.XXXs ] [ unserialize=X.XXXs ] [ wait write lock=X.XXXs ] [ manage broks=X.XXXs ] [ total=X.XXXs ]
```

Nature des broks traités

Affichage du type des **broks** traités : en quantité et en temps

```
[YYYY-MM-DD HH:MM:SS] INFO : [ WebUI-name ] [ MANAGE BROKS ] [PERF] => handled broks ->
count by types : [brok_type_1=XXXX] [brok_type_2=XX] [...]
```

```
[YYYY-MM-DD HH:MM:SS] INFO : [ WebUI-name ] [ MANAGE BROKS ] [PERF] => handled broks ->
time by types : [brok_type_1=XXXX] [brok_type_2=XX] [...]
```

Exemple de log

```
[2022-01-26 13:01:43] INFO : [ WebUI ] [ MANAGE BROKS ] [ PERF ] => handled broks -> count by types :
[initial_command_status=219] [update_broker_status=3] [update_program_status=1] [program_status=1]
[initial_contact_status=1] [...] [YYYY-MM-DD HH:MM:SS] INFO : [ WebUI ] [ MANAGE BROKS ] [ PERF ] => handled
broks -> time by types : [initial_command_status=0.022] [update_broker_status=0.000]
[update_program_status=0.000] [program_status=0.001] [initial_contact_status=0.000] [...]
```

L'absorption des broks a pris du retard

En cas de forte charge sur le serveur, ou lorsque des requêtes HTTP durent trop longtemps, le module peut prendre du retard sur la gestion des broks.

L'algorithme d'absorption des broks peut être paramétré via les paramètres **webui_broks_getter_XXX** du [fichier de configuration du Module WebUI](#)

Le mode de rattrapage pour récupérer les broks en retard s'active

Activation du rattrapage des broks en retard, on prend un **brok set** supplémentaire à traiter, on affiche :

- le nombre de **broks** dans le **brok set**
- le temps passé pour récupérer le **brok set** sur la queue
- le nombre actuel de **broks** à traiter
- le nombre maximal de **broks** qu'on peut récupérer avant de les traiter
- le nombre de **brok set** encore en attente

```
[YYYY-MM-DD HH:MM:SS] INFO : [ WebUI-name ] [ MANAGE BROKS ] [PERF] [LATE BROKS SETS] Getting brok set
with XX broks in X.XXXs [time for read queue size=X.XXXs]. Total broks to process= XXX/max:XXXX. Broks sets
in queue: X.
```

Le mode rattrapage a suffisamment de broks à traiter

Rattrapage des broks en retard en cours, on a atteint/dépassé le nombre maximal de broks à récupérer, on les traite :

```
[YYYY-MM-DD HH:MM:SS] INFO : [ WebUI-name ] [ MANAGE BROKS ] [PERF] [LATE BROKS SETS] Late brok taken =>
limit reach : XX / limit: XXXXXX.
```

Après avoir traité des broks, il en reste encore trop en attente

Après avoir traité des **broks**, il reste trop de **brok set** en attente, on garde le lock et on continue l'absorption des **broks** en retard :

```
[YYYY-MM-DD HH:MM:SS] INFO : [ WebUI-name ] [ MANAGE BROKS ] [PERF] Broks sets in queue after manage broks
is XX. We keep the lock and continue the brok managing.
```

Demande des broks initiaux lors du redémarrage d'un module externe du Broker

Lors du redémarrage d'un module externe du broker, une demande est envoyée par le Broker aux Schedulers pour récupérer de nouveaux broks initiaux (*une demande par Scheduler*).

```
[YYYY-MM-DD HH:MM:SS] INFO : [ broker-name ] [ GET BROKS ] [ NEED DATA ] [ scheduler-name ] I ask
for a initial broks generation to the scheduler with new daemon incarnation {u'shard_id': XXXX,
u'configuration_incarnation_uuid': UUID} (old incarnation was {})
```

Les logs du module MongoDB

Erreurs

Si le module MongoDB n'arrive pas à se connecter à la base mongo défini dans son fichier cfg :

```
[YYYY-MM-DD HH:MM:SS] ERROR : [ WebUI-name ] Mongoddb Module: Error : [ WebUI-name ] [ MONGODB ]  
- mongo connection failure to 192.168.1.87:27017
```

La configuration des Graphite backends

Port du Graphite backend invalide

L'adresse d'un graphite_backends contient un port non correct et est remplacé par le port par défaut (80) :

```
[YYYY-MM-DD HH:MM:SS] WARNING:[ WebUI-name ] [ CONFIGURATION ] The Graphite backend [ BACKEND ] is incorrect  
: The port [ INVALID_PORT ] is not valid. Valid values are integers from 0 to 65535.
```

Adresse du Graphite backend vide

Lorsqu'il n'y a pas d'adresse fournie dans un Graphite backend (*exemple : France::8080*), son adresse est remplacée par une adresse par défaut (0.0.0) :

```
[YYYY-MM-DD HH:MM:SS] WARNING:[ WebUI-name ] [ CONFIGURATION ] The Graphite backend [ BACKEND ] is  
incorrect : The hostname or IP address is empty or not found.
```

Backend mal formé

Lorsqu'un backend est mal formé, par exemple si il n'a pas de royaume et de port renseigné, un message est remonté au démarrage de la WebUI.

Exemple de graphite_backends erroné :

```
graphite_backends          192.168.1.23
```

```
[YYYY-MM-DD HH:MM:SS] ERROR : [ WebUI-name ] [ CONFIGURATION ] The Graphite backend [ BACKEND ] is not  
well formatted. It needs at least a realm and a host : <REALM>:<HOSTNAME>
```

Rappel de format attendu : <REALM>:<ADDRESS>:<PORT>

Exemple : France:192.168.1.23:8080

Protocole du Backend invalide

Si dans la définition du Backend, le protocole fourni n'est pas valide, un log au démarrage nous en averti.

Exemple de graphite_backends erroné :

```
graphite_backends          France:htt://192.168.1.23:8080
```

```
[YYYY-MM-DD HH:MM:SS] ERROR : [ WebUI-name ] [ CONFIGURATION ] The Graphite backend [ BACKEND ] is  
incorrect : The [ PROTOCOL ] protocol is unknown.
```

Les logs de communication avec Graphite

Lorsque la WebUI a besoin d'informations de métrologie, comme pour l'affichage de graphiques (*dans les rapports part exemple*), des informations sur les communications avec le serveur Graphite sont logguées :

Impossible de joindre le serveur Graphite

Si un des serveurs Graphite met trop de temps à répondre, alors ce message apparaît pour nous avertir que la prochaine tentative de connexion sera délayée (pour éviter de surcharger le réseau) :

```
[YYYY-MM-DD HH:MM:SS] ERROR : [ WebUI-name ] [ METROLOGY ] [ graphite:GRAPHITE_URL ] Graphite server currently unavailable. Will not try to recontact it until X seconds
```

Pas de serveur Graphite trouvé

Lorsque ce log s'affiche, il est possible que le module WebUI n'ai pas d'adresse **valide** de serveur Graphite. Il faut alors aller vérifier dans son fichier de configuration : **/etc/shinken/modules/webui.cfg**

```
[YYYY-MM-DD HH:MM:SS] ERROR : [ WebUI-name ] [ METROLOGY ] No valid Graphite backends found
```

Les log des appels de performances

Note : ces logs sont désactivés par défaut voir la page : Activation/Désactivation des parties de log pour les activer.

```
[YYYY-MM-DD HH:MM:SS] INFO : [ WebUI-name ] [ UI MANAGEMENT ] [ QUERY ] [ PERF ] [ NOM_DE_L'APPEL ] [ user UUID= UUID_DE_L'UTILISATEUR QUI_A_FAIT_LA_REQUETE ] [ start= 15:18:48 end= 15:18:48 Total= 0.000s { lock wait= 0.000s running time= 0.000s } ]
```

log de performance de la liste

Note ce log s'affichera en DEBUG par défaut et en WARNING si l'appel à la liste prend plus de 1s

```
[YYYY-MM-DD HH:MM:SS] WARNING : [ WebUI-name ] [ CP Server Thread-74 ] [ user=user-uuid ] [ get_data_visualisation_list ] [ PERF ] [ X.XXXs ] elements:[ in broker= XX filtered= XX total= XX in page= XX ] page:[ 1 / 1 ] filter:[ ] sort:[ ]
```

Les logs du module webui

Création des index en base de données au démarrage

Au démarrage du module, les index permettant d'assurer de bonnes performances pour les requêtes à la base de données sont créés s'ils n'existent pas.

Le temps pris pour la mise en place de chaque index est également détaillé.

```
[YYYY-MM-DD HH:MM:DD] INFO : [ WebUI-name ] [ Index ] Need to ensure indexes are present in Mongodb ( 2 indexes )
[YYYY-MM-DD HH:MM:DD] INFO : [ WebUI-name ] [ Index ] 1 - COLLECTION_NAME1::FIELD2 ( INDEX_NAME ) was created/checked in X.XXXSs
[YYYY-MM-DD HH:MM:DD] INFO : [ WebUI-name ] [ Index ] 2 - COLLECTION_NAME2::FIELD1, FIELD2 ( INDEX_NAME ) was created/checked in X.XXXSs
[YYYY-MM-DD HH:MM:DD] INFO : [ WebUI-name ] [ Index ] All Mongodb indexes were created/checked in X.XXXs
```

Exemple

```
[2021-11-25 16:38:47] INFO : [ WebUI3 ] [ Index ] Need to ensure indexes are present in Mongodb ( 1 indexes )
[2021-11-25 16:38:47] INFO : [ WebUI3 ] [ Index ] 1 - dashboard::uuid ( uuid_1 ) was created/checked in 0.0005s
[2021-11-25 16:38:47] INFO : [ WebUI3 ] [ Index ] All Mongodb indexes were created/checked in 0.0005s
```

Cas d'erreur

Si une erreur survient lors de la tentative d'indexation, le module essaiera à nouveau lors de son prochain démarrage, et le log suivant est généré

```
[YYYY-MM-DD HH:MM:SS] WARNING: [ WebUI-name ] Mongodb ERREUR PYTHON
[YYYY-MM-DD HH:MM:SS] WARNING: [ WebUI-name ] Mongodb index building could not be done, will retry at next
restart
```

Exemple

```
[2021-11-25 16:38:47] WARNING: [ WebUI3 ] Mongodb ERROR stack : Traceback (most recent call last):
[2021-11-25 16:38:47] WARNING: [ WebUI3 ] Mongodb   File "/var/lib/shinken/modules/webui/module.py", line
379, in main
[2021-11-25 16:38:47] WARNING: [ WebUI3 ] Mongodb       raise IOError
[2021-11-25 16:38:47] WARNING: [ WebUI3 ] Mongodb IOError
[2021-11-25 16:38:47] WARNING: [ WebUI3 ] Mongodb index building could not be done, will retry at next
restart
```

Erreurs lors du lancement du module WebUI

Le port de la WebUI est déjà ouvert

Si une autre WebUI utilise déjà le port (sûrement un problème de configuration), alors on aura les WARNING suivants:

```
[YYYY-MM-DD HH:MM:SS] WARNING: [ WebUI-name ] [TRY 1/10] The webui named [ WebUI-name ] can not start
because the address 0.0.0.0:7767 is already in use
[YYYY-MM-DD HH:MM:SS] WARNING: [ WebUI-name ] [TRY 2/10] The webui named [ WebUI-name ] can not start
because the address 0.0.0.0:7767 is already in use

[YYYY-MM-DD HH:MM:SS] WARNING: [ WebUI-name ] [TRY 3/10] The webui named [ WebUI-name ] can not start
because the address 0.0.0.0:7767 is already in use

[YYYY-MM-DD HH:MM:SS] WARNING: [ WebUI-name ] [TRY 4/10] The webui named [ WebUI-name ] can not start
because the address 0.0.0.0:7767 is already in use

[YYYY-MM-DD HH:MM:SS] WARNING: [ WebUI-name ] [TRY 5/10] The webui named [ WebUI-name ] can not start
because the address 0.0.0.0:7767 is already in use

[YYYY-MM-DD HH:MM:SS] WARNING: [ WebUI-name ] [TRY 6/10] The webui named [ WebUI-name ] can not start
because the address 0.0.0.0:7767 is already in use

[YYYY-MM-DD HH:MM:SS] WARNING: [ WebUI-name ] [TRY 7/10] The webui named [ WebUI-name ] can not start
because the address 0.0.0.0:7767 is already in use

[YYYY-MM-DD HH:MM:SS] WARNING: [ WebUI-name ] [TRY 8/10] The webui named [ WebUI-name ] can not start
because the address 0.0.0.0:7767 is already in use

[YYYY-MM-DD HH:MM:SS] WARNING: [ WebUI-name ] [TRY 9/10] The webui named [ WebUI-name ] can not start
because the address 0.0.0.0:7767 is already in use
```

Puis lors du dernier essai une ERROR (le module s'arrête):

```
[YYYY-MM-DD HH:MM:SS] ERROR : [ WebUI-name ] [ CRASH - INSIDE MODULE PROCESS ] [TRY 10/10] The webui named [
WebUI-name ] can not start because the address 0.0.0.0:7767 is already in use
```

Enfin l'erreur sera rapportée par le Broker qui va s'assurer que le module est éteint, et tenter de le relancer plus tard:

```
[YYYY-MM-DD HH:MM:SS] ERROR : [ broker-name ] [ MODULES-MANAGER ] [ MODULE-INSTANCE-CRASH ] [ WebUI-name ] [ module_type=webui ] The module WebUI2 just stopped. Last ERROR received:
```

```
[YYYY-MM-DD HH:MM:SS] ERROR : [ broker-name ] [ MODULES-MANAGER ] [ MODULE-INSTANCE-CRASH ] [ WebUI-name ] [ module_type=webui ] [TRY 10/10] The webui named [ WebUI-name ] can not start because the address 0.0.0.0:7767 is already in use
```

Erreurs issues d'un problème de changement dans le code de la WebUI (problème de DEV), ou d'une installation cassée si chez un client

Si le fichier index.html est cassé chez un client, ou qu'un développeur a changé ce fichier sans faire attention, on aura des erreurs spécifiques.

Si le fichier index.html est manquant:

```
[YYYY-MM-DD HH:MM:SS] ERROR : [ WebUI-name ] [ CRASH - INSIDE MODULE PROCESS ] The file /var/lib/shinken/modules/webui/htdocs/ui/index.html is missing: there is a critical error with your installation. Please open a ticket to your support.
```

Si le fichier index.html n'a pas les bons droits (l'utilisateur shinken ne peut pas l'ouvrir):

```
[YYYY-MM-DD HH:MM:SS] ERROR : [ WebUI-name ] [ CRASH - INSIDE MODULE PROCESS ] Cannot open the file /var/lib/shinken/modules/webui/htdocs/ui/index.html with the error "ERROR": there is a critical error with your installation. Please open a ticket to your support.
```

Si le fichier index.html n'a pas la bonne variable (soit erreur du DEV, soit le fichier du client a été modifié pour une raison)

```
[YYYY-MM-DD HH:MM:SS] ERROR : [ WebUI-name ] [ CRASH - INSIDE MODULE PROCESS ] The __shinken_lang__ variable was not found in the file /var/lib/shinken/modules/webui/htdocs/ui/index.html: there is a critical error with your installation. Please open a ticket to your support.
```

Erreurs de paramétrage

Si certains paramètres sont mal définis, la WebUI ne peut pas démarrer et va s'arrêter sur une erreur critique, qui sera affichée dans le check du Broker ainsi que dans le healthcheck.

Si son paramètre lang n'est pas dans la liste autorisé (fr, en), on aura l'erreur suivante:

```
[YYYY-MM-DD HH:MM:SS] ERROR : [ WebUI-name ] [ CRASH - INSIDE MODULE PROCESS ] For the parameter "lang" the value "XXX" is not allowed. Values can be : "fr, en"
```